01 Jan 2003

An Enhanced Least-squares Approach for Reinforcement Learning

Hailin Li

Cihan H. Dagli

Missouri University of Science and Technology, dagli@mst.edu

An Enhanced Least-Squares Approach for Reinforcement Learning

Hailin Li and Cihan H. Dagli

Department of Engineering Management
Smart Engineering Systems Laboratory
229 Engineering Management
University of Missouri-Rolla
Rolla MO 65409-0370
{hl8p5, dagli}@umr.edu

Abstract-This paper presents an enhanced Least-Squares approach for solving reinforcement learning control problems. Model-free Least-Squares policy iteration (LSPI) method has been successfully used for this learning domain. Although LSPI is a promising algorithm that uses linear approximator architecture to achieve policy optimization in the spirit of Q-learning, it faces challenging issues in terms of the selection of basis functions and training samples. Inspired by orthogonal Least-Squares regression (OLSR) method for selecting the centers of RBF neural network, we propose a new hybrid learning method. The suggested approach combines LSPI algorithm with OLSR strategy and uses simulation as a tool to guide the "feature processing" procedure. The results on the learning control of Cart-Pole system illustrate the effectiveness of the presented scheme.

I. INTRODUCTION

Reinforcement learning (RL) methods focus on the rational decision-making process under uncertain environments. The goal of RL is to analyze how decisions ought to be made in the light of clear objectives so that agent can generate a series of actions to influence the evolution of a stochastic dynamic system. The salient ability of RL to handle model-free situation makes RL more flexible than traditional dynamic programming. Ample research already has been done in this area. Among them the major development includes the temporal-difference learning algorithm proposed by Sutton [1] and Q-learning introduced in the thesis of Watkins [2].

As Kaelbling, Littman, and Moore [3] clearly illustrated, the intractability of solutions to sequential decision problems caused by the very large state-action space and the overwhelming requirement for computation presents a challenging array of difficulties in the reinforcement learning area. Such difficulties stimulate the development of approximation methods. Most of RL algorithms fit into the value function approximation category. Instead of approximating policies directly, the objective here is to select a parameterization of value function and then try to compute parameters that can produce an accurate approximation to the optimal value function. At present, linear function approximators are popular options as the value function approximation architecture mainly due to their transparent structure.

Unlike the Neuro-dynamic programming methods that require long time off-line simulation and training, S.Bradtke and A.Barto [4] introduced the linear Least-Squares algorithms for temporal difference learning (LSTD) and showed that LSTD can converge faster than conventional temporal difference learning methods in terms of the prediction ability. Unfortunately Koller and Parr [5] pointed out that LSTD could not be used directly as part of a policy iteration algorithm in many cases. In order to extend the linear Least-Squares idea to control problem, Lagoudakis and Parr [6] developed the model-free Least-Squares Q-learning (LSQ) and Least-Squares Policy Iteration (LSPI) algorithm. These algorithms produced good results on a variety of learning domains. The impressive aspects of LSPI include the effective sample data reusing, no approximate policy function needed and fast policy search speed if the "good" sample data set is selected. Similar to any linear approximator architecture, LSPI also faces the challenge of choosing basis functions. In essence, the feature extraction process needs lot of prior intuition about the problem. Furthermore, LSPI algorithm is very sensitive with the distribution of training samples. It produces the key disadvantage for applications.

Today, Radial Basis Function Neural Network (RBF NN) is used commonly as linear parameterization architecture, which is characterized by weighted combinations of basis functions. S.Chen, C.F.Cowan, and P.M.Grant [7] provided a systematic learning approach based on the orthogonal Least-Squares regression (OLSR) method to solve center selection problem so that the newly added center maximizes the amount of energy of the desired network output. This OLSR training strategy is a very efficient way for producing size-controllable RBF NN.

Motivated by the LSPI and OLSR training algorithm for RBF NN, a new enhanced Least-Squares learning method is proposed in this paper. Our effort is to produce the effective way to overcome the problem that LSPI algorithms [6] face, that is, selection of feature functions and sample data. In such a hybrid-learning scheme, a typical linear approximator using Gaussian function for all features is used as approximation architecture and the LSQ algorithm is used in order to approximate Q value functions. The number of features and the center of features are selected using OLSR training strategy based upon the training set generated by simulation. The proposed hybrid learning approach is applied to the classical Cart-Pole system and the simulation results are presented to show the effectiveness of the method.

0-7803-7898-9/03/$17.00 ©2003 IEEE

II. MDPS AND LEAST-SQUARES APPROXIMATION FOR Q-LEARNING

Our attention in this paper is restricted to discrete-time dynamic system that the system evolution at time t, action takes on a state x_t can be shown as

$$x_{t+1} = f(x_t, a_t, w_t), \tag{1}$$

where w_t is a disturbance and a_t is a control decision generated by policy μ. Each disturbance w_t is independently sampled from some fixed distributions.

In most reinforcement learning systems, its underlying control problem is modeled as a Markov Decision Process (MDP). The MDP can be denoted by a quadruple $\{S, A, P, R\}$ where: S is the state set, A is the action set, P is the state transition probability and R denotes the reward function $g(x_t, a_t)$. The policy μ is a mapping $\mu: S \to \Pr(A)$, where $\Pr(A)$ is a probability distribution in the action space. Let $\{x_0, x_1, x_2, ...\}$ be a Markov chain. For each policy μ, the value function J^μ is defined by

$$J^\mu(x) = E\left[\sum_{t=0}^{\infty} \alpha^t g(x_t, \mu(x_t)) \middle| x_0 = x\right], \tag{2}$$

where $\alpha \in [0,1)$ is a discount factor and state sequence is generated according to $x_0 = x$ and the system evolution.

J^* is used to represent the optimal value function. In Q-learning, Q value function is given by

$$Q^*(x,a) = \underset{w}{E}\left[g(x,a) + \alpha J^*(f(x,a,w))\right]. \tag{3}$$

Q value function is introduced to reduce computation requirement and the optimal actions can be obtained according to the following equation:

$$a_t = \arg\max_{a \in A} Q^*(x_t, a). \tag{4}$$

Q value functions can be stored in tables of size $|S||A|$, but it is not always the practical case for most real world applications. The intractability of state-action spaces calls for value function approximation. The LSQ algorithm [6] uses the parameterization of the linear form:

$$\hat{Q}(x,a,w) = \sum_{k=1}^{K} w(k)\phi_k(x,a) = \phi(x,a)^T W, \tag{5}$$

where $\phi_1, ..., \phi_k$ are "basis functions" generated through human intuition and trial-error process. Such functions extract the features of state-action space. $W = (w(1), ..., w(K))$ is a vector of scalar weights.

For a fixed policy μ,

$$\hat{Q}^\mu = \Phi W^\mu, \tag{6}$$

where Φ is $(|S||A| \times K)$ matrix and $K \ll |S||A|$.

If the model of MDP $(P^\mu, \Re)$ is available ($P^\mu(|S||A| \times |S||A|)$ denotes the transition probability from (x_t, a_t) to $(x_{t+1}, \mu(x_{t+1}))$), the form of Φ and the expected reward $\Re$ is:

$$\Phi = \begin{bmatrix} \phi(x_1, a_1)^T \\ \cdots \\ \phi(x, a)^T \\ \cdots \\ \phi(x_{|S|}, a_{|A|})^T \end{bmatrix}, \tag{7}$$

$$\Re = \begin{bmatrix} \sum_{x_{t+1}} P(x_1, a_1, x_{t+1}) g(x_1, a_1, x_{t+1}) \\ \cdots \\ \sum_{x_{t+1}} P(x, a, x_{t+1}) g(x, a, x_{t+1}) \\ \cdots \\ \sum_{x_{t+1}} P(x_{|S|}, a_{|A|}, x_{t+1}) g(x_{|S|}, a_{|A|}, x_{t+1}) \end{bmatrix}. \tag{8}$$

Under the model-free circumstance, LSQ uses the sample set collected from the MDP to construct the approximator. If samples $\{(x_i^t, a_i^t, r_i^t, x_i^{t+1}) | i = 1, 2, ..., L\}$ are "good" data set, which means that the selection of sample is potentially controlled by prior knowledge, the LSQ algorithm [6] $\hat{W}^\mu = \hat{A}^{-1}\hat{B}$ can promise the convergence to the true W^μ.

For the fixed policy μ, each sample (x^t, a^t, r^t, x^{t+1}) contributes to the construction of approximation (A, B matrix) by using:

$$\hat{A} \leftarrow \hat{A} + \phi(x^t, a^t)\left(\phi(x^t, a^t) - \gamma\phi(x^{t+1}, \mu(x^{t+1}))\right)^T, \tag{9}$$

$$\hat{B} \leftarrow \hat{B} + \phi(x^t, a^t) r^t, \tag{10}$$

where $\hat{A}(0) = 0$ and $\hat{B}(0) = 0$.

LSQ can learn state-action value functions of fixed policy effectively using the potentially controlled sample set so it is natural for [6] to extend the algorithm to policy iteration procedure, which is called Least-Squares Policy Iteration (LSPI) algorithm. Based upon the Q value function computed by LSQ, the next step optimal policy can be found simply using:

$$\mu^{t+1}(x) = \arg\max_a \hat{Q}^{\mu^t}(x,a) = \arg\max_a \phi(x,a)^T W^{\mu^t}. \tag{11}$$

The greedy policy is represented by the parameter W^{μ^t} and can be determined on demand for any given state. Clearly, the policy improvement procedure of LSPI can be

achieved without model knowledge and explicit representation for policy.

III. Simulation and Orthogonal Least-Squares Regression

Although it is reasonable to extend LSQ to control problems directly [8], failures are likely to happen for many applications mainly due to the significant bias for value approximation in the early steps. LSPI integrates the policy iteration idea and LSQ to solve learning control problems and produces more robust solution. The main reason for such result is that LSQ can use controllable sample set to approximate fixed policy. But the question still remains for the selection of basis functions for LSQ and "good" sample data sets.

The training samples of LSQ are collected from controllable "random episodes" starting from the very beginning. It is the source of approximation bias. The better sample data are, the faster approximation will converge to true value. Simulation is a powerful data generation tool for traditional neural network training, especially for the situation that system is hard to model but easy to simulate. Simulation can also tend to implicitly indicate the features of the system in terms of the state visiting frequency. This characteristic may help us to understand the potential useful system trajectories.

Orthogonal Least-Squares algorithm introduced by [7] for training an RBF network is a systematic learning approach for solving center selection problem so that the newly added center always maximizes the amount of energy of the desired network output. For the linear function approximators that have a single output, the network mapping can be viewed as a regression model of the form:

$$\begin{bmatrix} y(1) \\ y(2) \\ \dots \\ y(M) \end{bmatrix} = \begin{bmatrix} h(x_1,c_1,\sigma_1) & h(x_1,c_2,\sigma_2) & \dots & h(x_1,c_K,\sigma_K) \\ h(x_2,c_1,\sigma_1) & h(x_2,c_2,\sigma_2) & \dots & h(x_2,c_K,\sigma_K) \\ \dots & \dots & \dots & \dots \\ h(x_M,c_1,\sigma_1) & h(x_M,c_2,\sigma_2) & \dots & h(x_M,c_K,\sigma_K) \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \dots \\ w_K \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \\ \dots \\ e_M \end{bmatrix}.$$

Or, in matrix format: $Y = \mathrm{H}W + \mathrm{E}$.

The actual output of this RBF NN is given by

$$\hat{y} = \mathrm{H}\hat{W} = [\mathrm{h}_1, \ \mathrm{h}_2, \ \dots \ \mathrm{h}_K]\hat{W}. \qquad (12)$$

The centers of RBF NN are chosen from the input data set, which include M candidates. We summarize the algorithm [7] that performs the systematic selection of $K < M$ centers so that the size of RBF can be reduced significantly and the center of each basis function can be chosen by the order of their importance.

Step 1. $j = 1$, For $1 \le i \le M$,

$$b_1^{(i)} = \mathrm{h}_i,$$

$$[err]_1^i = \frac{\left(b_1^{(i)T} Y\right)^2}{b_1^{(i)T} b_1^{(i)} \cdot Y^T Y}.$$

Search

$$[err]_1^{i_1} = \max\left\{[err]_1^i, 1 \le i \le M\right\}.$$

Select

$b_1 = \mathrm{h}_{i_1}$, center $c_1 = c_{i_1}$.

Step 2. $j \ge 2$, For $1 \le i \le M$, $i \ne i_1, i \ne i_2, \dots, i \ne i_{j-1}$,

$$a_{pj}^i = \frac{b_p^T \mathrm{h}_i}{b_p^T b_p}, \quad 1 \le p \le j-1.$$

Let

$$b_j^i = \mathrm{h}_i - \sum_{p=1}^{j-1} a_{pj}^i b_p,$$

$$[err]_j^i = \frac{\left(b_j^{(i)T} Y\right)^2}{b_j^{(i)T} b_j^{(i)} \cdot Y^T Y}.$$

Search

$$[err]_j^{i_j} = \max\left\{[err]_j^i, 1 \le i \le M, i \ne i_1, i \ne i_2, \dots, i \ne i_{j-1}\right\}.$$

Select

$b_j = b_j^{i_j}$, center $c_j = c_{i_j}$.

Step 3. Repeat step 2. The algorithm is stopped at step N when $1 - \sum_{p=1}^{N} [err]_p \le \rho$, where $0 \le \rho \le 1$ is tolerance value defined by user.

IV. An Enhanced Least-Squares Approach in Reinforcement Learning for Control

Motivated by the simplicity of model-free LSPI algorithm and the effectiveness of OLSR method for selecting the centers of RBF NN among input data sets, a new hybrid-learning scheme for RL is proposed. The pre-learning process is added before Least-Squares policy iteration to set up parameters of LSPI. Such "feature processing" using OLSR provides a systematic way to select centers of basis functions for parameterization of linear form. It also guides the selection of sample data for LSPI.

As mentioned before, simulation is a powerful tool not only in the neural network communities but also for "on-line" reinforcement learning methods. Unlike the situation of those neural network applications, there is no readily available training set of input-output pairs that can be used to configure the feature of system in the RL context. For proposed enhanced learning method, simulation and classical one-step Q-learning [2] running for limited steps are used for "feature processing" to roughly evaluate the state-action value functions under given policies. Although it will create analytical difficulties, this is probably the only way to extract the system characteristics under model-free circumstance.

Fig. 1 illustrates the operation of the enhanced Least-Squares method. We also provide the brief description for the complete algorithm.

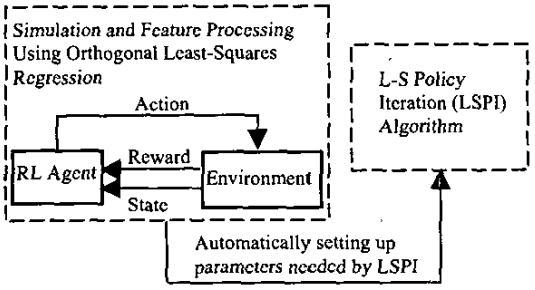

Fig. 1. Architecture of an enhanced Least-Squares Approach

Assuming that the problem can be formulated as kind of finite MDP. The simulation for model-free system starts from a random state following the greedy policy produced by preliminary Q-learning and generates a series of observable time sequence sample set $\Gamma:\left\{\left(x_i^t,a_i^t,r_i^t,x_i^{t+1}\right)\middle| i=1,2,...,L\right\}$, where $x_i \in S$, and $a_i \in A$. The total action types of $a \in A$ are N.

Step 1. Initialize the table representation for complete Q value set, $Q\left(x_{|S|},a_{|A|}\right)=0$.

Step 2. While the stop criterion is not satisfied (this step will be terminated far before the classic Q-learning algorithm converge to true value)

a. Calculate the approximate state-action values $Q(x\in\Gamma,a\in\Gamma)$ using one-step Q-learning algorithm.
b. *Simulation will follow the greedy policy.* At present, the strong exploration ability is preferred so the noise η_t is likely to take comparatively big value.

Step 3. Generate N input-output training data set $\left\{(X_d \leftrightarrow Y_d)\middle| d=1,2,...N\right\}$ from simulation set Γ for center selection, where $X_d=\left\{(x\in\Gamma,a_d\in\Gamma)\right\}$ and $Y_d=Q(X_d)$.

Step 4. Using orthogonal Least-Squares regression algorithm [7] described in section III to select N kinds of centers set for LSQ's basis functions from N training set. Each selected centers set will be used to approximate the value functions for corresponding action type.

Step 5. Refine simulation sample set Γ. It is reasonable to remove obviously useless samples base upon rough Q-values at step 2 so that the bias will likely be decreased.

Now we set up the parameters for LSPI as following:

$k_d\middle| d=1,2,...N$: Number (center) of basis functions for state-action$_{(d|d=1,2...N)}$ space.

ϕ : Using Gaussian function as LSQ basis function.

Γ : Training sample set for LSQ.

Step 6. Let $W_0=0$, $W^{\mu}=W_0$ and $W^{\mu'}=W^{\mu}$ (Initial policy).

Do

Get sample data from Γ (Add/remove/maintain samples)

$\mu=\mu'$, that is, $W^{\mu}=W^{\mu'}$

$\mu'=LSQ\left(\Gamma,k,\phi,W^{\mu}\right)$, Compute $W^{\mu'}$

While $\left(\mu\neq\mu'\right)$.

The new insight of the above enhanced Least-Squares approach for RL is the "feature processing" procedure in which simulation is used as a tool to produce the collection of samples for LSQ and orthogonal Least-Squares center selection algorithm is used to generate the basis functions for LSQ algorithm and guide the selection for "good" samples. The effectiveness of the proposed method will be illustrated in the classical Cart-Pole problem.

V. Results in the Cart-Pole System

The classic cart and pole dynamic system is used to assess the proposed enhanced L-S approach. The objective of the problem is to exert a sequence of forces upon the cart's center of mass so that the pole is balanced for as long as possible and the cart does not hit the end of the track. For the RL *controller, the dynamics of this system is assumed to be* unknown. But in our simulation case following dynamics described by [9] is used for the system.

$$\ddot{\theta}_t=\frac{g\sin\theta_t+\cos\theta_t\left[\frac{-F_t-m_p l\dot{\theta}_t^2\sin\theta_t+\mu_c\,\mathrm{sgn}\left(\dot{x}_t\right)}{m_c+m_p}\right]-\frac{\mu_p\dot{\theta}_t}{m_p l}}{l\left[\frac{4}{3}-\frac{m_p\cos^2\theta_t}{m_c+m_p}\right]} \tag{13}$$

and

$$\ddot{x}_t=\frac{F_t+m_p l\left[\dot{\theta}_t^2\sin\theta_t-\ddot{\theta}_t\cos\theta_t\right]-\mu_c\,\mathrm{sgn}\left(\dot{x}_t\right)}{m_c+m_p}. \tag{14}$$

In the experiments, the parameters $x_t,\dot{x}_t,\theta_t,\dot{\theta}_t,m_c,m_p,\mu_c,\mu_p,l,F_t$ are the horizontal position of the cart relative to the track, the horizontal velocity of the cart, the angle between the pole and vertical, clockwise being positive, the angular velocity of the pole, the mass of the cart (1.0), mass of the pole (0.1), coefficient of friction of cart on track, coefficient of friction of pivot, the distance from center of mass of pole to the pivot (0.5), and the force exerted on the cart's center of mass at time t, respectively. The sample frequency for system simulation and applying control force are the same (50 Hz). There are 2 possible action types here: $A=\{-10,+10\}$ but the actual force to the system is the noisy signal $F_t=(a_t+\eta_t)$, where $a_t\in A$ and noise η_t follows

uniformly distribution. The state at time t is specified by variables $\{x_t, \dot{x}_t, \theta_t, \dot{\theta}_t\}$ and the continuous state space is separated to 163 discrete states. The external reinforcement signal (reward) is defined as:

$$r_t = \begin{cases} 0, & \text{if -0.21 radians} < \theta_t < 0.21 \text{ radians and -2.4m} < x_t < 2.4\text{m} \\ -1, & \text{otherwise.} \end{cases}$$

The results are shown in following two figures. Simulation starts from a random state and follows rough policy learned from preliminary one-step Q-learning. The selected state-action centers for basis functions are plotted in Fig. 2. Orthogonal Least-Squares center selection algorithm is applied to two input-output training data sets generated by simulation. A set of 70 Gaussian functions (35 for each action type) over one dimension state space is generated automatically without human involvement to approximate the state-action value functions. The selection result is also used to refine the sample set Γ for LSQ, which means we are likely to remove the data far away from the selected centers. Fig. 3 illustrates the performance of controller learned by three different RL methods, that is: classical one-step Q-learning, LSPI and proposed hybrid Least-Squares method. In order to show the results clearly, the Y-axis of figure shows the log function value for successful balancing time at each training episode. After only about 350 training episodes, the proposed method returns the policy under which system-balancing period already exceeds 5000 seconds (250000 steps). Obviously such convergence speed is much faster than one-step Q-learning and a little bit better than simple LSPI. Considering the fact that for simple LSPI there is much human-based tweaking work needs to be done, our enhanced Least-Squares method for RL is more robust and human-independent.

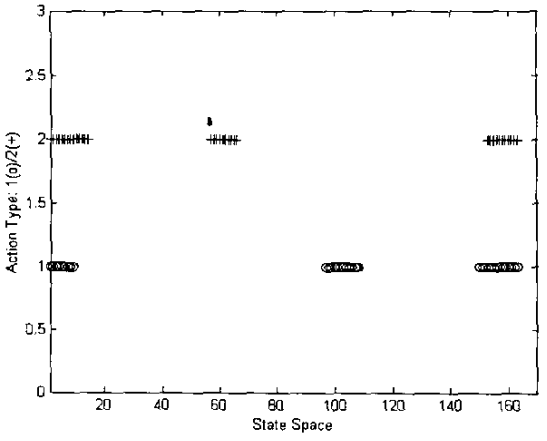

Fig. 2. Selected State-Action Centers

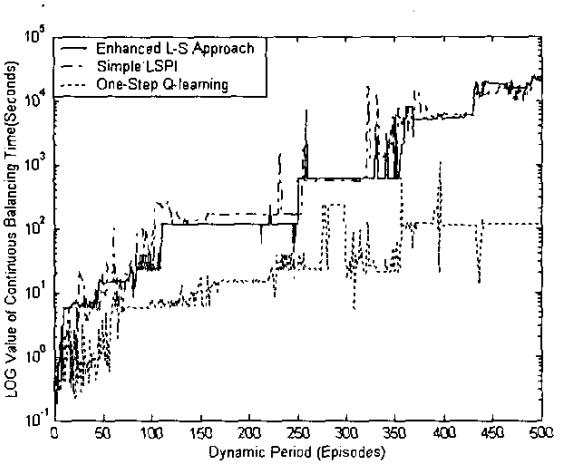

Fig. 3. Results by using different Reinforcement Learning methods

VI. Conclusions

A novel enhanced Least-Squares learning method is proposed in this paper to solve reinforcement learning control problems. The method combines Least-Squares policy iteration algorithm with OLS regression strategy that can select system feature centers automatically. Simulation is introduced as a tool to generate rough representation for feature so that the construction of training sample set for model-free Least-Squares Q-learning is guided. The simulation experiments on the Cart-Pole system demonstrate the effectiveness of the proposed hybrid approach. Other applications of this method are now in progress.

References

[1] R.S. Sutton, "Learning to predict by the methods of temporal difference," Machine Learning, Vol.3, No.1, 1988, pp. 9-44.

[2] C.J.C.H. Watkins, "Learning from delayed rewards," PhD thesis, Cambridge University, Cambridge, UK, 1989.

[3] Kaelbling, L.P., Littman, M.L., and Moore, A.W., "Reinforcement Learning: a survey," Journal of Artificial Intelligence Research, 4, 1996, pp. 237-285.

[4] Steven J.Bradtke and A.Barto, "Linear Least-Squares algorithms for temporal difference learning," Machine Learning, 22(1/2/3), 1996, pp. 33-57.

[5] Daphne Koller and Ronald Parr, "Policy iteration for factored MDPs," Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (UAI-00), Morgan Kaufmann, 2000, pp. 326-334.

[6] Michail Lagoudakis and Ronald Parr, "Model free Least Squares policy iteration," Proceedings of the 14th Neural Information Processing Systems (NIPS-14), Vancouver, Canada, Dec., 2001.

[7] S.Chen, C.F.Cowan, and P.M.Grant, "Orthogonal Least Squares algorithm for Radial Basis Function Networks," IEEE Transactions on Neural Networks, vol.21, 1990, pp. 2513-2539.

[8] Michail Lagoudakis and Michael L.Littman, "Algorithm selection using Reinforcement Learning," Proceedings of the 7th International Conference on Machine Learning, San Francisco, CA, 2000, pp. 511-518.

[9] R.S. Sutton, "Temporal aspects of credit assignment in Reinforcement Learning", PhD thesis, University of Massachusetts, 1984.